Autenticação HTTP

HTTP fornece uma estrutura geral para controle de acesso e autenticação. Esta página é uma introdução à estrutura HTTP para autenticação e mostra como restringir o acesso ao seu servidor usando o esquema "Basic" do HTTP.

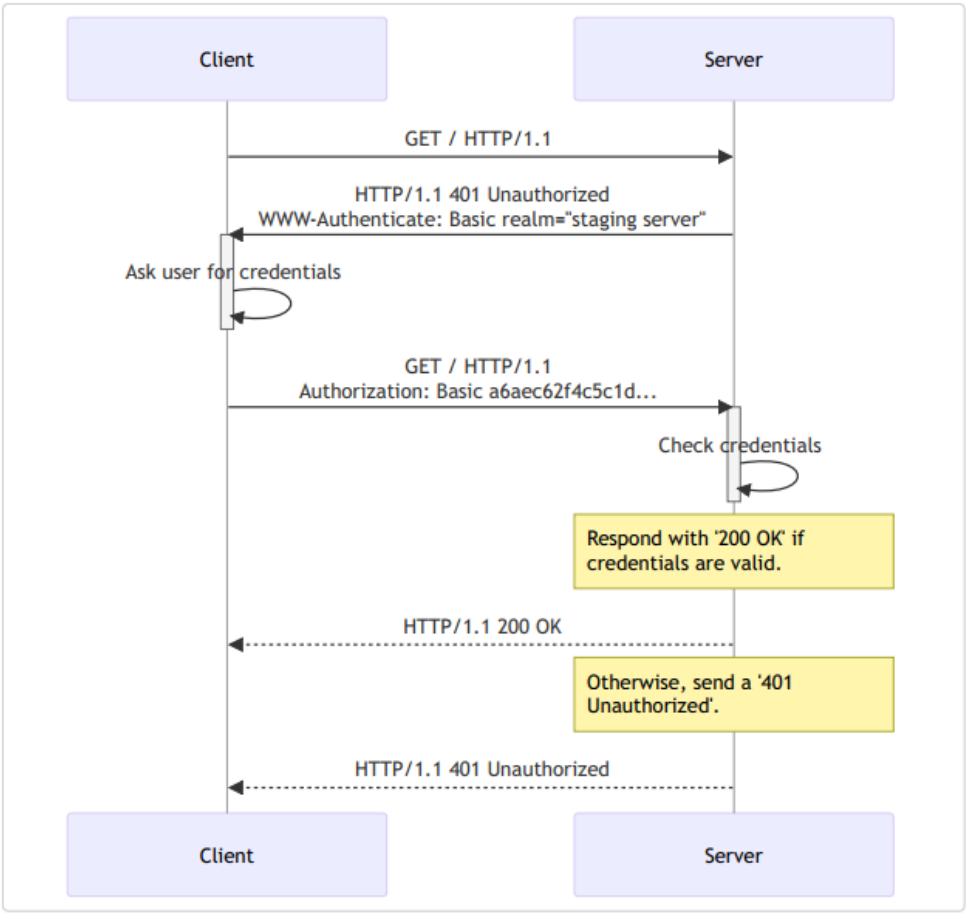
A estrutura geral de autenticação HTTP

O RFC 7235 define a estrutura de autenticação HTTP, que pode ser usada por um servidor para desafiar uma requisição do cliente, e por um cliente para fornecer informações de autenticação.

O fluxo de desafio e resposta funciona assim:

1. O servidor responde a um cliente com um status de resposta **401 (Unauthorized)** e fornece informações sobre como se autorizar com um cabeçalho de resposta WWW-Authenticate contendo pelo menos um desafio.
2. Um cliente que deseja se autenticar com o servidor pode então fazê-lo incluindo um cabeçalho de requisição Authorization com as credenciais.
3. Normalmente, um cliente apresentará um prompt de senha ao usuário e então emitirá a requisição incluindo o cabeçalho Authorization correto.

Fluxo de mensagens:



O fluxo geral de mensagens acima é o mesmo para a maioria (se não todos) dos esquemas de autenticação. As informações reais nos cabeçalhos e a forma como são codificadas mudam!

⚠️ **Aviso**: O esquema de autenticação "Basic" usado no diagrama acima envia as credenciais codificadas, mas não criptografadas. Isso seria completamente inseguro a menos que a troca ocorra sobre uma conexão segura (HTTPS/TLS).

Autenticação via proxy

O mesmo mecanismo de desafio e resposta pode ser usado para autenticação de proxy. Como tanto autenticação de recurso quanto de proxy podem coexistir, é necessário um conjunto diferente de cabeçalhos e códigos de status. No caso de proxies:

- O código de status de desafio é **407 (Proxy Authentication Required)**
- O cabeçalho de resposta é Proxy-Authenticate, contendo pelo menos um desafio aplicável ao proxy
- O cabeçalho de requisição é Proxy-Authorization, usado para fornecer as credenciais ao servidor proxy.

Acesso proibido

Se um servidor (ou proxy) recebe credenciais inválidas, ele deve responder com um **401 Unauthorized** ou com um **407 Proxy Authentication Required**, e o usuário pode enviar uma nova requisição ou substituir o campo de cabeçalho Authorization.

Se um servidor (ou proxy) recebe credenciais válidas mas que são inadequadas para acessar um determinado recurso, o servidor deve responder com o código de status **403 Forbidden**. Diferente de 401 ou 407, a autenticação é impossível para esse usuário, e os navegadores não proporão uma nova tentativa.

Em todos os casos, o servidor pode preferir retornar um código de status **404 Not Found**, para ocultar a existência da página de um usuário sem privilégios adequados ou não autenticado corretamente.

Autenticação de imagens cross-origin

Embora imagens possam ser exibidas em páginas web de diferentes origens (domínios), o navegador **não enviará** credenciais (como cookies ou cabeçalhos Authorization) a menos que o atributo crossorigin seja definido.

Se você quiser carregar uma imagem com autenticação, adicione o atributo crossorigin ao seu elemento HTML:

```html
<img src="https://example.com/image.png" crossorigin />
```

Codificação de caracteres

Credenciais no esquema Basic são transmitidas como um valor username:password, codificado com **Base64**. Isso significa que você deve usar apenas **ASCII**, já que caracteres fora da faixa ASCII não são corretamente interpretados por todos os servidores.

ℹ️ **Nota**: Embora o RFC 7617 afirme que codificações não ASCII **podem** ser usadas (por exemplo, UTF-8 com Base64), muitos servidores não dão suporte adequado a isso. Portanto, recomenda-se manter nomes de usuário e senhas usando apenas o conjunto de caracteres ASCII.

Cabeçalhos relacionados

WWW-Authenticate

Este é o cabeçalho de resposta que indica o(s) método(s) de autenticação aceitos pelo servidor. Ele é enviado com uma resposta 401 Unauthorized (ou 407 Proxy Authentication Required para proxies).
Por exemplo:

```http
WWW-Authenticate: Basic realm="Secure Area"
```

Pode haver múltiplos cabeçalhos WWW-Authenticate, um para cada método de autenticação aceito.

Authorization

Este é o cabeçalho de requisição onde o cliente envia suas credenciais ao servidor, após receber o desafio de autenticação via WWW-Authenticate.

Formato típico para autenticação básica:

```http
Authorization: Basic <credentials>
```

Onde <credentials> é a string username:password codificada em Base64.

Proxy-Authenticate e Proxy-Authorization

Esses são os equivalentes de WWW-Authenticate e Authorization, mas usados para autenticação de **proxies** HTTP.

Esquemas de autenticação

A especificação HTTP define vários esquemas de autenticação que podem ser usados com WWW-Authenticate e Authorization. Os navegadores modernos atualmente dão suporte principalmente a dois deles:

- **Basic**
- **Bearer**

Outros esquemas de autenticação como Digest, HOBA, Mutual, AWS4-HMAC-SHA256 e SCRAM foram definidos, mas **não são suportados pelos navegadores**.

Basic

O esquema **Basic** é amplamente compatível com clientes e servidores HTTP, e o mais simples de implementar.

- Ele codifica o par username:password com **Base64** (sem criptografia).
- Essa codificação é transmitida em **texto claro**, e portanto deve **sempre** ser usada com **HTTPS** para segurança.
- Um prompt de login típico do navegador será exibido quando um recurso protegido por autenticação básica for acessado.

Exemplo:

```http
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

Aqui, a string Base64 representa: Aladdin:open sesame

Bearer

O esquema **Bearer** é normalmente usado com **OAuth 2.0**. Em vez de enviar um nome de usuário e senha, o cliente envia um **token de acesso**:

```http
Authorization: Bearer mF_9.B5f-4.1JqM
```

Este token representa as permissões concedidas ao cliente (por exemplo, acesso a recursos de um usuário autenticado) e é emitido por um servidor de autorização. Tokens Bearer são amplamente usados em APIs RESTful.

⚠️ **Aviso**: Ambos os esquemas Basic e Bearer são **vulneráveis à interceptação** se forem usados sobre conexões não seguras. O uso de **HTTPS** é essencial.

Exemplos

Usando autenticação básica com o curl

```bash
curl -u username:password https://example.com/
```

Isso envia uma requisição HTTP com um cabeçalho Authorization contendo as credenciais codificadas em Base64.

Usando autenticação bearer com o curl

```bash
curl -H "Authorization: Bearer YOUR_TOKEN" https://example.com/
```

Substitua YOUR_TOKEN pelo token de acesso real fornecido por um provedor de autenticação OAuth 2.0.